CS 9644 – SCALABLE AND RELIABLE DISTRIBUTED SYSTEMS

REQUIRED GRADUATE INDIVIDUAL WRITE-UP TOPIC: *MapReduce*

ASIF IHTEMADUL HAQUE

ID: 251462715

EMAIL: ahaque62@uwo.ca

Google’s MapReduce framework, introduced in a 2004 paper by Jeff Dean and Sanjay Ghemawat, revolutionized distributed computing by simplifying the processing of large datasets across commodity hardware. It addressed the complexity and redundancy of writing custom distributed code by providing a simple yet powerful programming model with **Map** and **Reduce** functions.

Understanding MapReduce

Map Function

The Map function converts input data (usually text) into a set of key-value pairs, like counting word occurrences in a dataset. Each line of text is an input, identified by its line number and content. The mapper function outputs intermediate key-value pairs, where keys are words and values are counts.

Reduce Function

The Reduce function then takes these sorted intermediate pairs from all mappers, grouped by key, and aggregates them to generate the final output.

Implementation Overview

Job Execution

1. **Input Splitting**: Input files stored on Google's Distributed File System (GFS) are split into smaller chunks (typically 16–64 MB each). Each chunk is assigned to a map task.

2. **Mapping**: Tasks are scheduled near the nodes with input data to reduce network bandwidth usage. Each mapper processes its assigned input chunk, generates intermediate key-value pairs, and locally sorts and partitions them.

3. **Partitioning**: Intermediate pairs are partitioned into **R** partitions based on a partition function, typically a hash of the key. Each partition is prepared for one reducer.

4. **Shuffling**: Reducer tasks remotely fetch their corresponding partitions from all mapper nodes. This transfer is known as “shuffling."

5. **Reducing**: Reducers merge the sorted partitions and apply the reduce function, producing the final sorted output. This output is then written to GFS with replication for fault tolerance.

6. **Job Completion**: Upon completion, the master node notifies the client program that the MapReduce job is done.

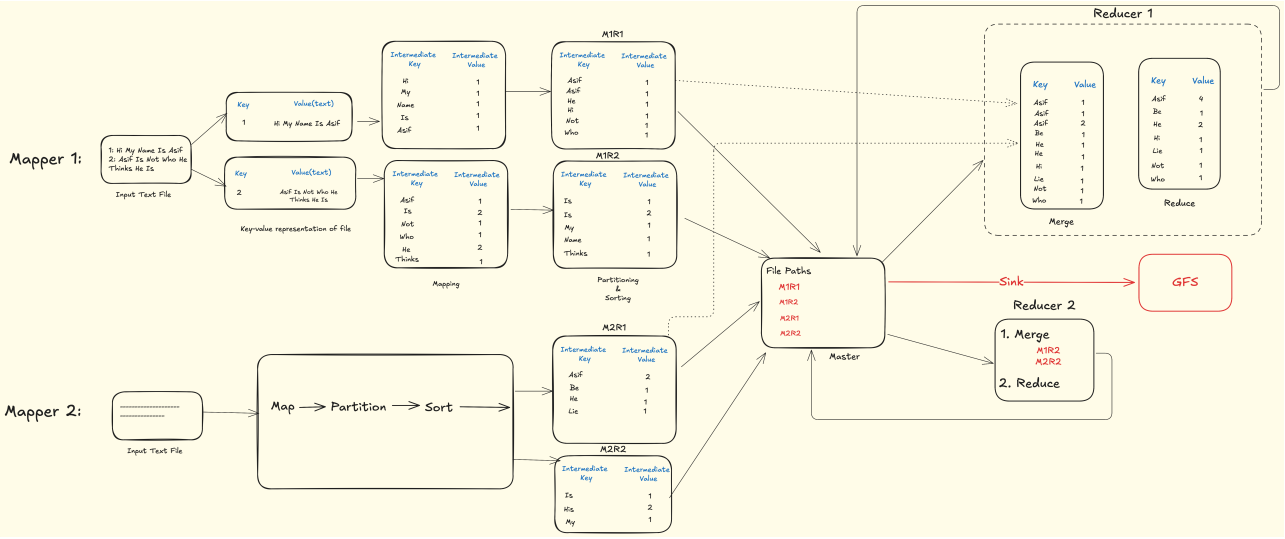

Fig: MapReduce Workflow

Fault Tolerance

- **Mapper Failures**: If a mapper node fails, its tasks are rescheduled on other nodes. Intermediate results from failed mappers are discarded, and the tasks are recomputed.
- **Reducer Failures**: Failed reducer tasks are similarly rescheduled and recomputed.
- **Master Failures**: While rare, master node failures lead to job termination and require the job to be restarted entirely by the client.

Performance Optimization

- **Local Writes:** Mappers store their output locally rather than immediately replicating to GFS, conserving network bandwidth.
- **Combiner Functions:** Reducer functions may be executed locally on mapper nodes (as combiners) to reduce intermediate data size before shuffling.
- **Straggler Handling:** To handle slow nodes, MapReduce proactively schedules redundant task executions near completion, reducing job latency by eliminating bottlenecks from slow nodes.

Impact

Google’s MapReduce model streamlined development, leading to rapid adoption and over 900 jobs created in its first year. Its simple programming model enabled efficient batch processing, fault tolerance, and scalability, paving the way for frameworks like Apache Hadoop and Spark.